

# Design Document

(Draft version 4)

## 1. INTRODUCTION

### 1.1 Goals

- **An event driven job schedule**
  - ◆ Simulator scans the event sequence and do the operation related to every event in time order.
  - ◆ Event can be job submit/job finish, monitor event or other event added by the user.
  - ◆ An overall method invokes and initializes all the modules and the handles of the modules will be transported into the simulator.
  - ◆ The simulator should be able to support other modules and their subclasses.
- **A user command line interface**
  - ◆ User can pass all the parameters by command line
  - ◆ Advantage user interface can be used to call the command line entry automatically.
  - ◆ A system parameter config file can be used to initialize the command line parameter
  - ◆ A file name config file can be used to initialize all the temp, debug and output path, name and extension name.
  - ◆ The data read from the config file are on low level, so the parameter given in the command line will replace the same data read in config file.
- **Extendable module design**
  - ◆ These modules should have the standard interface.
  - ◆ All modules are supposed to know all the data formats. Hence, they can get correct data from the dictionary type of parameter. And any modification in data format should be specified clearly in the design document.
  - ◆ The modules can be extended in 2 ways: subclass and new method.
  - ◆ Also, new function can be added to the existed method. But this kind of modification should be static, which is used in all extension.
- **Running time interface**
  - ◆ Keep receiving running time information and show them in the user friendly way.
- **Result analysis and show**
  - ◆ Read job trace result file and do the statistics as request.
  - ◆ Show the analysis result in graph.
  - ◆ New graph method can be added to it easily.
- **Input and output files**
  - ◆ Input raw files: Job trace and Node structure files

- ◆ Formatted files: Job trace, Node structure, Job and Node config files
- ◆ Output Result files: Job simulator result, Event log and Debug log.

## 2. STRUCTURE

### 2.1 Function Map

The program contains 5 parts:

User Interface	
cqsim	<ul style="list-style-type: none"> <li>• Basic user command line interface.</li> <li>• All parameters should be transferred by command line.</li> <li>• Additional profile is allowed, but corresponding explain program should be designed.</li> </ul>
filter	<ul style="list-style-type: none"> <li>• Job and node filter command line interface.</li> <li>• Call the filter process to read raw files and output the data into the formatted file.</li> <li>• Also provide a port to output the formatted data list.</li> </ul>
cqsim_ad	<ul style="list-style-type: none"> <li>• Advanced user interface, to simplify the user input.</li> <li>• Parameters are stored in a profile.</li> <li>• Can be designed as a command line interface that user need to only provide the profile file name, or a graphic user interface.</li> <li>• Call the basic command line interface <b>cqsim</b> with the data.</li> </ul>
cqsim_main	<ul style="list-style-type: none"> <li>• Define all modules and transfer these modules to the simulator .</li> <li>• Different modules can be chosen here.</li> <li>• Call the simulator <b>Cqsim_sim</b>, transfer the modules(in a dictionary data) and parameters into the simulator.</li> <li>• Start the simulation process.</li> <li>• Import the path file Cqsim_path.py.</li> </ul>
cqsim_path	<ul style="list-style-type: none"> <li>• Contain all path value</li> <li>• Be invoked if the file need to access other file in some other place</li> </ul>
Result_analysis	<ul style="list-style-type: none"> <li>• Call the result analysis program to deal with the result.</li> </ul>
<b>Modules</b>	

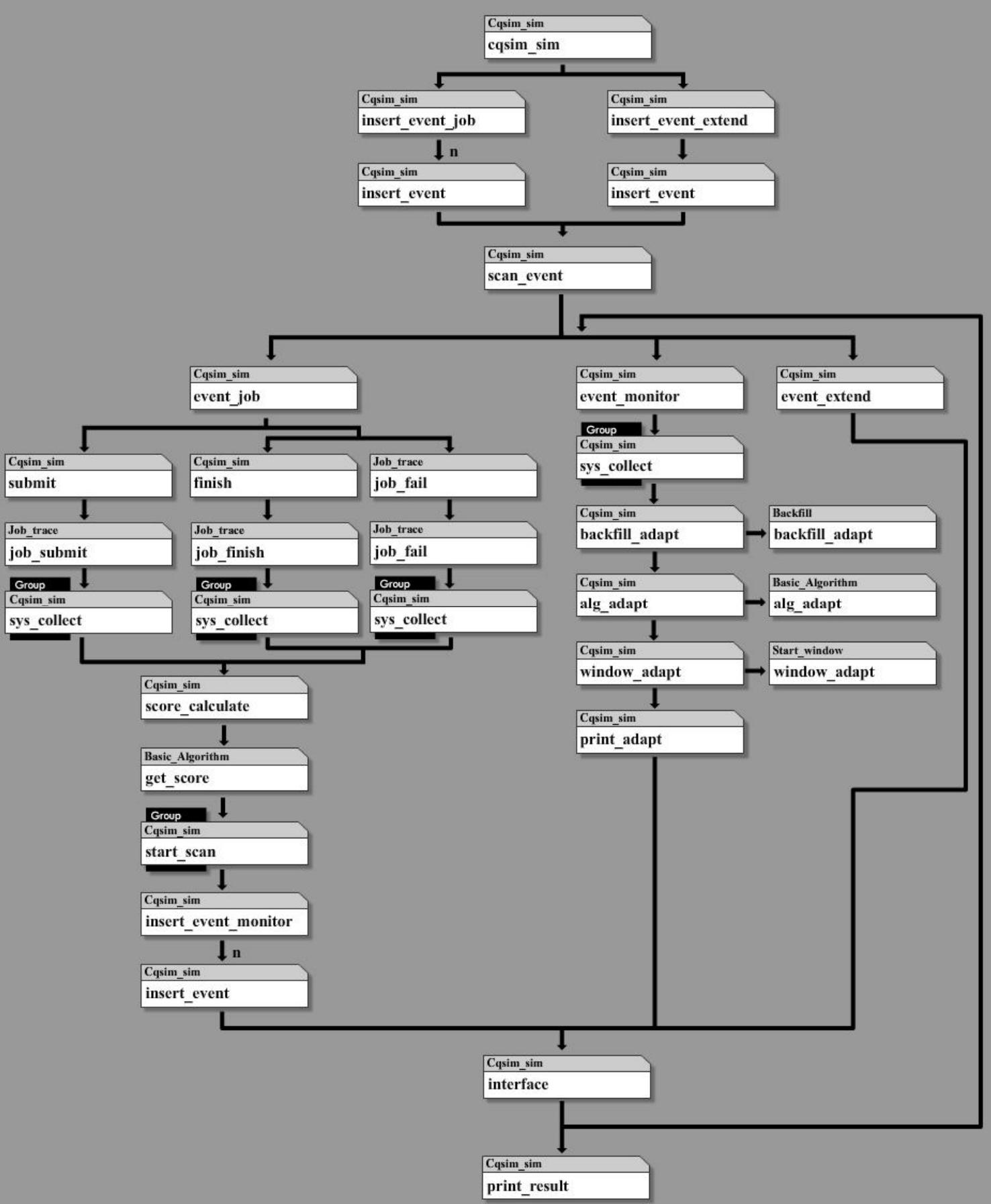
	<ul style="list-style-type: none"> <li>• All the modules should contain: <code>__init__()</code>, <code>reset()</code> method to initialize and reset the basic setting.</li> <li>• At least one interface for other module to call it with the input running time parameters.</li> </ul>
<b>Filter_job</b>	<ul style="list-style-type: none"> <li>• Receive job trace file name and other parameters.</li> <li>• Read the file and extract the necessary information.</li> <li>• Format the data according to the parameters and store them into a list.</li> <li>• Store the data into a temp file according to the parameters.</li> <li>• Store the overall job trace information into a config file.</li> <li>• Provide output port to transfer the formatted data.</li> </ul>
<b>Filter_node</b>	<ul style="list-style-type: none"> <li>• Receive node structure file name and other parameters.</li> <li>• Read the file and extract the necessary information.</li> <li>• Format the data according to the parameters and design and store them into a list.</li> <li>• Store the data into a temp file according to the parameters.</li> <li>• Store the overall node structure information into a config file.</li> <li>• Provide output port to transfer the formatted data.</li> </ul>
<b>Job_trace</b>	<ul style="list-style-type: none"> <li>• Receive formatted job trace file name or the formatted job trace data.</li> <li>• Read the temp file and store the data into a list.</li> <li>• Provide all the job trace operations, and keep tracing the information of every job.</li> </ul>
<b>Node_struct</b>	<ul style="list-style-type: none"> <li>• Receive formatted node structure file name or the formatted node structure data.</li> <li>• Read the temp file and store the data into a list.</li> <li>• Provide all the node structure operations, and keep tracing the information of every node.</li> <li>• Provide the prediction of the state of the node structure.</li> <li>• Provide the function to check the prediction data.</li> </ul>
<b>Backfill</b>	<ul style="list-style-type: none"> <li>• Receive parameters when it is initialized.</li> <li>• Provide backfill operation: receive the current state of the waiting list, make some prediction by calling the <b>node structure object</b>, return the index list of the jobs which can be backfilled now.</li> <li>• Different backfill mode can be added by designing a new backfill method and build the relationship between mode number and backfill function in <b>main()</b> method</li> <li>• Adapt function can be called by the simulator, to modify the</li> </ul>

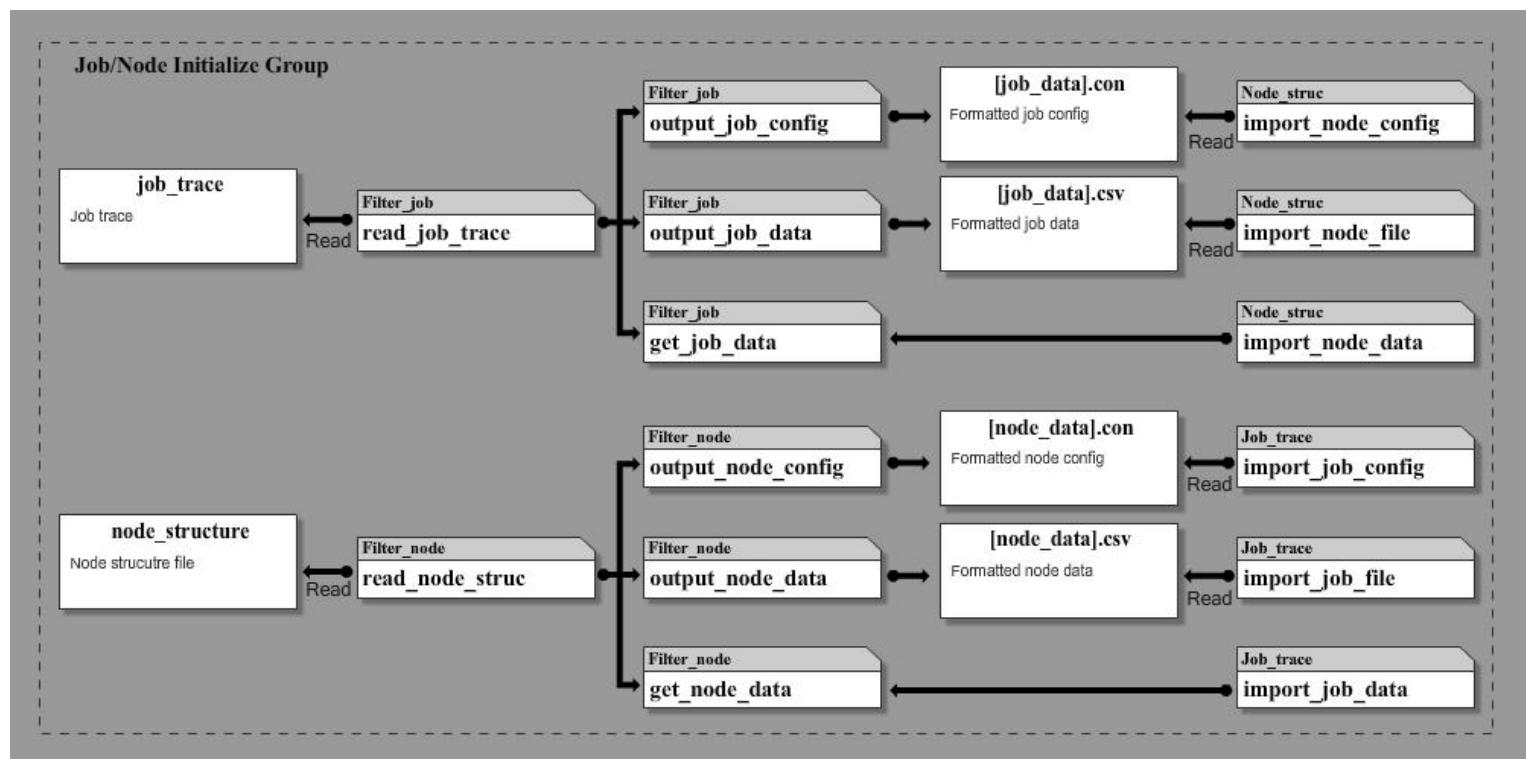
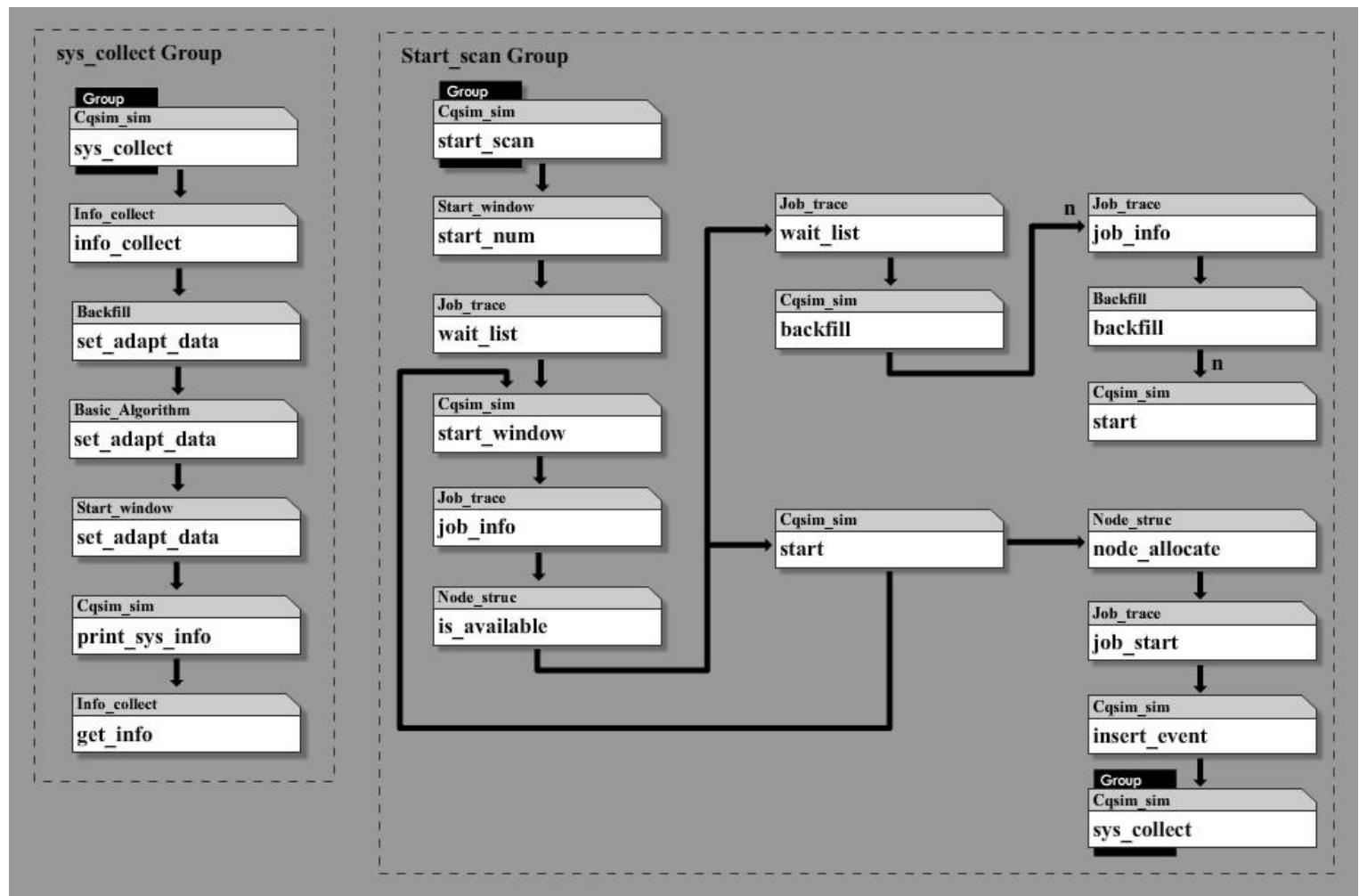
	<p>parameters in running time depend on the changing of system state.</p> <ul style="list-style-type: none"> <li>▪ Adapt config file name is transmitted into the module in adapt parameter list.</li> <li>▪ All the adapt parameters and the requested average utilization interval list are get from the config file.</li> <li>▪ Provide a method to analysis and set the adapt value in the <b>info_collect</b> module</li> <li>▪ Check the most new system information in <b>info_collect</b> module to see whether it reach the adapt request. Call the adapt method if so.</li> </ul> <p>• Extend: User can also design a subclass of it if the current backfill structure can not reach the request. If you do so, import the right subclass in <b>cqsim_main()</b> method, and modify the input running-time parameters in <b>backfill()</b> method in <b>Cqsim_sim</b> class. Also, you may want to modify the initial parameters in <b>cqsim_main()</b> method and re-design the command line in both <b>Cqsim()</b> method and <b>Cqsim_ad()</b> method. So does the corresponding config files.</p>
<b>Start_window</b>	<ul style="list-style-type: none"> <li>• Receive parameters when it is initialized.</li> <li>• Provide window operation when look for the next job to start: Receive <b>x</b> job indexes with related system information which need to be scanned in waiting list, Change the order of the waiting jobs according to the window function. Then return the new order. The simulator will call the window operation again when <b>y</b> job has started after the last window operation in one event iteration. Provide port to output <b>x</b> and <b>y</b></li> <li>• This module will reorder the waiting list before any job starts in this iteration.</li> <li>• Different window mode: Similar to <b>Backfill</b> module</li> <li>• Adapt function: Similar to <b>Backfill</b> module</li> <li>• Extend: Similar to <b>Backfill</b> module</li> </ul>
<b>Bacis_Algorithm</b>	<ul style="list-style-type: none"> <li>• Receive parameters when it is initialized.</li> <li>• Receive algorithm list and assemble the elements into an algorithm string.</li> <li>• Receive the information of a job and return the job score. Also can receive a list of job information and then return the</li> </ul>

		<p>corresponding list of scores in the same order.</p> <ul style="list-style-type: none"> <li>• Adapt function: Similar to <b>Backfill</b> module</li> <li>• Extend: Similar to <b>Backfill</b> module</li> </ul>
	<b>Info_collect</b>	<ul style="list-style-type: none"> <li>• Collect all the system information for record and analysis.</li> <li>• Provide collect and read operations. Hence other methods can check and store the information.</li> </ul>
	<b>Log_print</b>	<ul style="list-style-type: none"> <li>• Provide all the output file operation for the simulator.</li> <li>• Result, running time information and debug log can be done by invoking this module.</li> <li>• Provide the basic operation on files: open, write and close.</li> <li>• Changing style of log can be done by design a different subclass of it.</li> <li>• Every <b>Log_print</b> object can only manage a file in one time.</li> </ul>
	<b>Debug_log</b>	<ul style="list-style-type: none"> <li>• Receive the debug level: <ul style="list-style-type: none"> <li>0: No debug</li> <li>1-3: Three debug level, 3 is the highest.</li> <li>4: Print the debug information on the screen.</li> <li>5,6: Print the method and module name.</li> </ul> </li> <li>• User should provide the debug log content with the level number.</li> <li>• The debug module will print the given content depending on the input level number.</li> </ul>
	<b>Output_log</b>	<ul style="list-style-type: none"> <li>• Provide 3 output log print method: <ul style="list-style-type: none"> <li>System information log</li> <li>Job result log</li> <li>Adapt information log</li> </ul> </li> <li>• System information log and Adapt information log method are invoked in every iteration.</li> <li>• Job result log is printed when all jobs are done.</li> </ul>
	Simulator	<ul style="list-style-type: none"> <li>• Receive parameters and module handles.</li> <li>• Contain an inside event sequence, every event information includes virtual time, event type, event priority and event parameter list.</li> <li>• The simulator can add, delete or modify the event sequence in running time.</li> <li>• There are 3 kinds of event: job event(Job submit/finish), monitor event and extend event which is specially designed for new requirement.</li> <li>• Job submit events added to the sequence before all the process.</li> </ul>

	<p>Monitor events (from time A to B) added to the sequence when a job starts at A and finish at B. If there exist same monitor event at one time point, no new monitor event will be added.</p> <p>Job finish event added when the job start.</p> <p>User designed event added depending on the design.</p> <ul style="list-style-type: none"><li>• In running time, simulator move its virtual time from one event to the next, and stop when all events are done and no more new event comes.</li><li>• Simple flow of the 3 kinds of event:<ul style="list-style-type: none"><li>◆ Job event - job start scan - system information collect</li><li>◆ Monitor event - adapt function</li><li>◆ Extend event - user designed function</li></ul></li><li>• Call the run-time interface to show the running time state after every event</li><li>• Print system information log at every event.</li></ul> <p>Output job result file when all jobs are done.</p>
	<p><u>Run-time Interface</u></p> <ul style="list-style-type: none"><li>•</li></ul>
	<p><u>Result Analysis</u></p> <ul style="list-style-type: none"><li>•</li></ul>

## 2.2 Flow Diagram





### 3. Module

#### 3.1 Overall

This is a sample.

Name	<i>Method name</i>			
<b>Input</b>	<i>Parameter Name</i>	<i>(type)</i>	<i>Initial value</i>	<i>Comment</i> <i>The parameter is necessary if it has no initial value</i>
<b>Output</b>	<i>Return value type</i>	<i>(type)</i>		<i>Comment</i>
<b>Process</b>	<ul style="list-style-type: none"> <li><i>Detail of the duty of the method</i></li> </ul>			

#### 3.2 Filter job

Name	<u>__init__</u>			
<b>Input</b>	trace	(string)	-	Path and name of the job trace file.
	save	(string)	None	Path and name of the format job trace file which the formatted job trace data will be stored in.
	config	(string)	None	Path and name of the format job trace config file
	sdate	(date)	None	The date and time of the first selected job. If it is None, no modification will be made.
	start	(float)	-1	Virtual submit time of the first selected job._j
	density	(float)	1.0	The scale of the submit time of the job trace. The virtual submit time will be: [(Original submit time - first job submit time + start) * density]
	anchor	(int)	0	The index of the first job will be read in the job trace file.
	rnum	(int)	0	The number of jobs will be read.
	debug	(handle)	None	Debug module handle
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Initialize the parameters.</li> </ul>			

Name	<u>reset</u>			
<b>Input</b>	trace	(string)	None	-
	save	(string)	None	-

	config	(string)	None	-
	sdate	(date)	None	-
	start	(float)	None	-
	density	(float)	None	-
	anchor	(int)	None	-
	rnum	(int)	None	-
	debug	(handle)	None	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Reset the parameters.</li> </ul>			

<b>Name</b>	reset_config_data			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Reset config data</li> </ul>			

<b>Name</b>	read_job_trace			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Open the job trace file with path string <b>[trace]</b></li> <li>Read the job trace file and store <b>[rnum]</b> jobs starting at <b>[anchor]</b> position.</li> <li>Modify the start date of the selected job trace to <b>[start]</b> if it is not None.</li> <li>Modify the submit time of the jobs: [(Original submit time - first job submit time + <b>start</b>) * <b>density</b>]</li> <li>Formatted all the selected job data and store them into a local list.</li> </ul>			

<b>Name</b>	input_check			
<b>Input</b>	jobInfo	(dictionary)	-	Input job data
<b>Output</b>	(int)	(int)		1 for correct, <0 for error
<b>Process</b>	<ul style="list-style-type: none"> <li>Check the input job data.</li> <li>Return negative number if any error found.</li> </ul>			

<b>Name</b>	get_job_data			
<b>Input</b>	None	-	-	-
<b>Output</b>	(list)	(list)		Return the formatted job trace data
<b>Process</b>	<ul style="list-style-type: none"> <li>Return the formatted job trace data without other additional information</li> </ul>			

<b>Name</b>	get_job_num			
-------------	-------------	--	--	--

<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	( int )		Return the length of the formatted job list
<b>Process</b>	<ul style="list-style-type: none"> <li>Return the length of the formatted job list.</li> </ul>			

<b>Name</b>	output_job_data			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Open the formatted job data file with path [save]</li> <li>Store the list and other information in the designed format.</li> </ul>			

<b>Name</b>	output_job_config			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Open the formatted job config file with path [config]</li> <li>Store the overall job config data</li> </ul>			

### 3.3 Filter node

<b>Name</b>	<u>__init__</u>			
<b>Input</b>	struc	(string)	-	Path and name of the node structure file
	save	(string)	None	Path and name of the temp node structure file which the formatted node structure data will be stored in.
	config	(string)	None	Path and name of the format node structure config file
	debug	(handle)	None	Debug module handle
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Initialize the parameters.</li> </ul>			

<b>Name</b>	reset			
<b>Input</b>	struc	(string)	None	-
	save	(string)	None	-
	config	(string)	None	-
	debug	(handle)	None	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Reset the parameters.</li> </ul>			

<b>Name</b>	reset_config_data			
-------------	-------------------	--	--	--

<b>Input</b>	None	-	-	-	
<b>Output</b>	None	-			-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Reset config data</li> </ul>				

<b>Name</b>	read_node_struc				
<b>Input</b>	None	-	-	-	
<b>Output</b>	None	-			-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Open the node structure file with path string <b>[struc]</b></li> <li>• Formatted the node structure and store them into a local list.</li> </ul>				

<b>Name</b>	input_check				
<b>Input</b>	nodeInfo	(dictionary)	-		Input node data
<b>Output</b>	(int)	(int)			1 for correct, <0 for error
<b>Process</b>	<ul style="list-style-type: none"> <li>• Check the input node data.</li> <li>• Return negative number if any error found.</li> </ul>				

<b>Name</b>	get_node_num				
<b>Input</b>	None	-	-	-	
<b>Output</b>	(int)	(int)			Return the length of the formatted node list
<b>Process</b>	<ul style="list-style-type: none"> <li>• Return the length of the formatted node list.</li> </ul>				

<b>Name</b>	get_node_data				
<b>Input</b>	None	-	-	-	
<b>Output</b>	(list)	(list)			Return the formatted node structure data.
<b>Process</b>	<ul style="list-style-type: none"> <li>• Return the formatted node structure data without other additional information</li> </ul>				

<b>Name</b>	output_node_data				
<b>Input</b>	None	-	-	-	
<b>Output</b>	None	-			-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Open the formatted node structure file with path <b>[save]</b></li> <li>• Store the list and other information in the designed format.</li> </ul>				

<b>Name</b>	output_node_config				
<b>Input</b>	None	-	-	-	

<b>Output</b>	None	-	/\	-
<b>Process</b>	<ul style="list-style-type: none"> <li>Open the formatted node config file with path <b>[config]</b></li> <li>Store the overall node config data</li> </ul>			

### 3.4 Job trace

<b>Name</b>	<u>__init__</u>			
<b>Input</b>	start	(float)	-1	Virtual submit time of the first selected job._j
	num	(int)	0	The number of jobs will be read.
	anchor	(int)	0	The index of the first job will be read in the job trace file.
	density	(float)	1.0	The scale of the submit time of the job trace. The virtual submit time will be: [(Original submit time - first job submit time + start) * density]
	debug	(handle)	None	Debug module handle
<b>Output</b>	None	-	/\	-
<b>Process</b>	<ul style="list-style-type: none"> <li>Initialize the parameters.</li> </ul>			

<b>Name</b>	reset			
<b>Input</b>	start	(float)	None	-
	num	(int)	None	-
	anchor	(int)	None	-
	density	(float)	None	-
	debug	(handle)	None	-
<b>Output</b>	None	-	/\	-
<b>Process</b>	<ul style="list-style-type: none"> <li>Reset the parameters.</li> </ul>			

<b>Name</b>	import_job_file			
<b>Input</b>	job_file	(string)	-	Path and name of the formatted temp job data file.
<b>Output</b>	None	-	/\	-
<b>Process</b>	<ul style="list-style-type: none"> <li>Open the temp job data file with path string <b>[job_file]</b></li> <li>Store the information into the local buffers.</li> </ul>			

<b>Name</b>	import_job_config			
<b>Input</b>	config_file	(string)	-	Path and name of the formatted job config file.
<b>Output</b>	None	-	/\	-
<b>Process</b>	<ul style="list-style-type: none"> <li>Open the job config file with path string <b>[config_file]</b></li> <li>Store the config information into the local buffers.</li> </ul>			

<b>Name</b>	import_job_data			
<b>Input</b>	job_data	(list)	-	Formatted job trace data list.
<b>Output</b>	None	-	\diagdown	-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Store the income job data into the local list.</li> </ul>			

<b>Name</b>	submit_list			
<b>Input</b>	None	-	-	-
<b>Output</b>	(list)	(list)	\diagdown	Return the job list which have not been submitted.
<b>Process</b>	<ul style="list-style-type: none"> <li>• Return the job list which have not been submitted.</li> </ul>			

<b>Name</b>	wait_list			
<b>Input</b>	None	-	-	-
<b>Output</b>	(list)	(list)	\diagdown	Return the current waiting list.
<b>Process</b>	<ul style="list-style-type: none"> <li>• Return the current waiting list.</li> </ul>			

<b>Name</b>	run_list			
<b>Input</b>	None	-	-	-
<b>Output</b>	(list)	(list)	\diagdown	Return the current running list.
<b>Process</b>	<ul style="list-style-type: none"> <li>• Return the current running list.</li> </ul>			

<b>Name</b>	done_list			
<b>Input</b>	None	-	-	-
<b>Output</b>	(list)	(list)	\diagdown	Return the job list which are done.
<b>Process</b>	<ul style="list-style-type: none"> <li>• Return the job list which are done.</li> </ul>			

<b>Name</b>	wait_size			
<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	( int )	\diagdown	Return the total size of the waiting job
<b>Process</b>	<ul style="list-style-type: none"> <li>• Return the total size of the waiting job.</li> </ul>			

<b>Name</b>	refresh_score			
<b>Input</b>	score	(float)	-	The new score or score list (if [job_index] is None)

	job_index	(int)	None	The index of the selected job.
<b>Output</b>	(dictionary)	( dictionary )		Return the detail of the job indicated by the input index #.
<b>Process</b>	<ul style="list-style-type: none"> <li>Refresh the score of the selected job if index is given</li> <li>Refresh the scores of all jobs in the old order if no index is given.</li> <li>Reorder the wait list in the order of score (from high to low-)</li> </ul>			

<b>Name</b>	scoreCmp			
<b>Input</b>	jobIndex_c1	(int)	-	-
	jobIndex_c2	(int)	-	-
<b>Output</b>	<cmp>	<cmp>		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Method used to order.</li> </ul>			

<b>Name</b>	job_info			
<b>Input</b>	job_index	(int)	-1	The index of the selected job.
<b>Output</b>	(dictionary)	( dictionary )		Return the detail of the job indicated by the input index #.
<b>Process</b>	<ul style="list-style-type: none"> <li>Return the detail of the job.</li> <li>If job_index is -1, return the whole job trace information</li> </ul>			

<b>Name</b>	job_submit			
<b>Input</b>	job_index	(int)	-	The index of the selected job.
	job_score	(int)	0	The score of the selected job.
	job_est_start	(int)	-1	The estimated start time of the selected job.
<b>Output</b>	(int)	(int)		1: Success 0: Fail
<b>Process</b>	<ul style="list-style-type: none"> <li>Submit the selected job</li> <li>Move the submit pointer to the next job and add the index of the job to waiting list.</li> <li>Modify the state of the job from "not-submit" to "waiting".</li> <li>Fill other information of the job. (e.g. scores of the job)</li> <li>Return 0 if any error occurs. Otherwise return 1.</li> </ul>			

<b>Name</b>	job_start			
<b>Input</b>	job_index	(int)	-	The index of the selected job.
	time	(float)	-	Start time
<b>Output</b>	(int)	(int)		1: Success 0: Fail
<b>Process</b>	<ul style="list-style-type: none"> <li>Start the selected job</li> <li>Delete the index of the job from waiting list and add the index of the job to running list.</li> <li>Modify the state of the job from "waiting" to "running".</li> <li>Fill other information of the job. (e.g. start time)</li> <li>Return 0 if any error occurs. Otherwise return 1.</li> </ul>			

<b>Name</b>	job_finish			
<b>Input</b>	job_index	(int)	-	The index of the selected job.
	time	(float)	None	Finish time
<b>Output</b>	(int)	(int)		1: Success 0: Fail
<b>Process</b>	<ul style="list-style-type: none"> <li>Finish the selected job</li> <li>Delete the index of the job from running list and add the index of the job to done list.</li> <li>Modify the state of the job form "running" to "done".</li> <li>Fill other information of the job.</li> <li>Return 0 if any error occur. Otherwise return 1.</li> </ul>			

<b>Name</b>	job_fail			
<b>Input</b>	job_index	(int)	-	The index of the selected job.
	time	(float)	None	Finish time
<b>Output</b>	(int)	(int)		1: Success 0: Fail
<b>Process</b>	<ul style="list-style-type: none"> <li>Mark the selected job failed</li> <li>Delete the index of the job from running list and add the index of the job to fail list.</li> <li>Modify the state of the job form "running" to "fail".</li> <li>Fill other information of the job.</li> <li>Return 0 if any error occur. Otherwise return 1.</li> </ul>			

<b>Name</b>	job_set_score			
<b>Input</b>	job_index	(int)	-	The index of the selected job.
	score	(float)	-	The score of the selected job
<b>Output</b>	(int)	(int)		1: Success 0: Fail
<b>Process</b>	<ul style="list-style-type: none"> <li>Modify the score of the job</li> <li>Fill other information of the job.</li> <li>Return 0 if any error occur. Otherwise return 1.</li> </ul>			

### 3.5 Node struc

<b>Name</b>	__init__			
<b>Input</b>	debug	(handle)	None	Debug module handle
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Initialize the parameters.</li> </ul>			

<b>Name</b>	reset			
-------------	-------	--	--	--

<b>Input</b>	debug	(handle)	None	-	
<b>Output</b>	None	-		-	
<b>Process</b>	<ul style="list-style-type: none"> <li>Reset the parameters.</li> </ul>				

<b>Name</b>	read_list				
<b>Input</b>	source_str	(string)	None	The string need to be analysis into a list	
<b>Output</b>	(list)	(list)		The list get from the string	
<b>Process</b>	<ul style="list-style-type: none"> <li>Translate a string into a list of int</li> <li>The string must be like [a,b,...,z]</li> </ul>				

<b>Name</b>	import_node_file				
<b>Input</b>	node_file	(string)	-	Path and name of the formatted temp node data file.	
<b>Output</b>	None	-		-	
<b>Process</b>	<ul style="list-style-type: none"> <li>Open the temp node data file with path string <b>[node_file]</b></li> <li>Store the information into the local buffers.</li> </ul>				

<b>Name</b>	import_node_config				
<b>Input</b>	config_file	(string)	-	Path and name of the formatted node config file.	
<b>Output</b>	None	-		-	
<b>Process</b>	<ul style="list-style-type: none"> <li>Open the node config file with path string <b>[config_file]</b></li> <li>Store the config information into the local buffers.</li> </ul>				

<b>Name</b>	import_node_data				
<b>Input</b>	node_data	(list)	-	Formatted node structure data list.	
<b>Output</b>	None	-		-	
<b>Process</b>	<ul style="list-style-type: none"> <li>Store the income node data into the local list.</li> </ul>				

<b>Name</b>	is_available				
<b>Input</b>	proc_num	(int)	-	The amount of request processes	
<b>Output</b>	(int)	(int)		1: Yes 0: No	
<b>Process</b>	<ul style="list-style-type: none"> <li>Check whether the request process is available.</li> <li>Return 1 for available, 0 for not available.</li> </ul>				

<b>Name</b>	get_tot				
<b>Input</b>	None	-	-	-	

<b>Output</b>	(int)	(int)	/\	Return total processe number.
<b>Process</b>	• Return total processe number.			

<b>Name</b>	get_idle			
<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	(int)	/\	Return current idle processe number.
<b>Process</b>	• Return current idle processe number.			

<b>Name</b>	get_avail			
<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	(int)	/\	Return current max available idle processe number.
<b>Process</b>	• Return current max available idle processe number.			

<b>Name</b>	node_allocate			
<b>Input</b>	proc_num	(int)	-	Request processe number
	start	(float)	-	Current virtual time
	end	(float)	-	Job expect end time.
	job_index	(int)	-	The index of the job which requests the processe.
<b>Output</b>	(int)	(int)	/\	1: Success 0: Fail
<b>Process</b>	<ul style="list-style-type: none"> <li>Find the available processe and mark them with the <b>[job_index]</b>.</li> <li>Modify other information.</li> <li>Return 1 if every thing is OK, otherwise return 0.</li> </ul>			

<b>Name</b>	node_release			
<b>Input</b>	job_index	(int)	-	The index of the job which release the processe.
	end	(float)	-	Job end time.
<b>Output</b>	(int)	(int)	/\	1: Success 0: Fail
<b>Process</b>	<ul style="list-style-type: none"> <li>Release all the processe which marked as <b>[job_index]</b>.</li> <li>This method need at least 1 input parameter and the parameter should be identically named.</li> <li>Mark the released processe with "idle"</li> <li>Modify other related information</li> <li>Return 1 if every thing is OK, otherwise return 0.</li> </ul>			

<b>Name</b>	pre_avail			
<b>Input</b>	proc_num	(int)	-	The amount of request processe
	start	(float)	-	Current virtual time

	end	(float)	None	Job expect end time.
<b>Output</b>	(int)	(int)		1: Yes 0: No
<b>Process</b>	<ul style="list-style-type: none"> <li>Check whether the job can run from <b>[start]</b> to <b>[end]</b> with all the prediction.</li> <li>If <b>[end]</b> is None, then set it to <b>[start]</b></li> <li>Return 1 for available, 0 for not available.</li> </ul>			

	reserve			
<b>Input</b>	proc_num	(int)	-	The amount of request processes
	job_index	(int)	-	The index of the job which requests the processes.
	time	(float)	-	Job expect run time.
	start	(float)	None	Current virtual time
	index	(int)	-1	The index of the prediction list start to scan
<b>Output</b>	(int)	(int)		1: Yes 0: No
<b>Process</b>	<ul style="list-style-type: none"> <li>Reserve the job can from <b>[start]</b> to <b>[end]</b> in the prediction data.</li> <li>If <b>[start]</b> is None, just find a space to reserve it</li> <li>If <b>[index]</b> is -1, scan the prediction list from 0, otherwise scan from <b>[index]</b></li> <li>Return 1 for available, 0 for not available.</li> </ul>			

	pre_delete			
<b>Input</b>	proc_num	(int)	-	The amount of request processes
	job_index	(int)	-	The index of the job which requests the processes.
<b>Output</b>	(int)	(int)		1: Yes 0: No
<b>Process</b>	<ul style="list-style-type: none"> <li>Delete <b>[node_num]</b> number of processes from the reserved job whose index is <b>[job_index]</b></li> <li>Return 1 for available, 0 for not available.</li> </ul>			

	pre_modify			
<b>Input</b>	proc_num	(int)	-	The amount of request processes
	start	(float)	-	Current virtual time
	end	(float)	-	Job expect end time.
	job_index	(int)	-	The index of the job which requests the processes.
<b>Output</b>	(int)	(int)		1: Yes 0: No
<b>Process</b>	<ul style="list-style-type: none"> <li>Modify the reserve data of the selected job.</li> <li>Return 1 for available, 0 for not available.</li> </ul>			

	pre_get_last			
<b>Input</b>	None	-	-	-
<b>Output</b>	(dictionary)	(dictionary)		The dictionary contain the last value of all kind of information

<b>Process</b>	• Scan the prediction job list and return the last value of start and end time
----------------	--

<b>Name</b>	pre_reset			
<b>Input</b>	time	(int)	-	Current virtual time
<b>Output</b>	(int)	(int)	\diagdown	1: Success 0: No
<b>Process</b>	<ul style="list-style-type: none"> <li>Reset the prediction list</li> <li>Clean the prediction list, then scan the node state and build the initial prediction list.</li> </ul>			

<b>Name</b>	find_res_place			
<b>Input</b>	proc_num	(int)	-	The process number request
	index	(int)	-	The index of prediction list start to scan
	time	(int)	-	Current virtual time
<b>Output</b>	(int)	(int)	\diagdown	-1: Can reserve the job starting at [index] >=0: The index not available for the reservation
<b>Process</b>	<ul style="list-style-type: none"> <li>Scan the prediction list from [index], return the index of the position in prediction list where the is not available for the reservation. Otherwise, return -1</li> </ul>			

<b>Name</b>	find_place			
<b>Input</b>	proc_num	(int)	-	The process number request
<b>Output</b>	(list)	(list )	\diagdown	List of the allocated job index
<b>Process</b>	<ul style="list-style-type: none"> <li>Find the request node, return the list of node index</li> </ul>			

<b>Name</b>	recover_place			
<b>Input</b>	node_list	(list)	-	The node index lit need to release
<b>Output</b>	None	-	\diagdown	-
<b>Process</b>	<ul style="list-style-type: none"> <li>Release the node whose index are in the input list.</li> </ul>			

### 3.6 Backfill

<b>Name</b>	__init__			
<b>Input</b>	mode	(int)	0	Backfill mode, no difference will be made if only one mode designed.
	ad_mode	(int)	0	Adapt backfill mode
	node_module	(handle)	None	Node structure module handle
	info_module	(handle)	None	System information module handle
	debug	(handle)	None	Debug module handle

	para_list	(list)	None	Additional parameter.
	ad_para_list	(list)	None	Adapt parameter.
<b>Output</b>	None	-	\	-
<b>Process</b>	<ul style="list-style-type: none"> <li>Initialize the parameters.</li> <li>Initialize the adapt parameters.</li> </ul>			

<b>Name</b>	reset			
<b>Input</b>	mode	(int)	None	-
	ad_mode	(int)	None	-
	node_module	(handle)	None	-
	info_module	(handle)	None	-
	debug	(handle)	None	-
	para_list	(list)	None	-
	ad_para_list	(list)	None	-
<b>Output</b>	None	-	\	-
<b>Process</b>	<ul style="list-style-type: none"> <li>Reset the parameters.</li> <li>Reset the adapt parameters.</li> </ul>			

<b>Name</b>	backfill			
<b>Input</b>	wait_job	(list)	-	The list of the related waiting job with the details. Each job information is a dictionary.
	para_in	(dictionary)	None	Running time parameters in the dictionary type.
<b>Output</b>	(list)	( list )	\	List of the backfill jobs. None for no job can be backfill.
<b>Process</b>	<ul style="list-style-type: none"> <li>This is the entry of the backfill module.</li> <li>Receive the running time information and store them into the local buffers, then invoke <b>main</b> method to deal with the request.</li> <li>Get the first backfill job index(in wait list) from the <b>main</b> method and return it to the invoker.</li> </ul>			

<b>Name</b>	main			
<b>Input</b>	None	-	-	All the parameters should be stored in the local buffer.
<b>Output</b>	(list)	( list )	\	List of the backfill jobs. None for no job can be backfill.
<b>Process</b>	<ul style="list-style-type: none"> <li>Provide the backfill function.</li> <li>Return the List of index of the backfill jobs .</li> <li>It select different backfill mode by the input parameter <b>[mode]</b>, and invoke corresponding backfill method.</li> </ul>			

<b>Name</b>	backfill_EASY			
-------------	---------------	--	--	--

<b>Input</b>	None	-	-	All the parameters should be stored in the local buffer.
<b>Output</b>	(list)	( list )		List of the backfill jobs. None for no job can be backfill.
<b>Process</b>	<ul style="list-style-type: none"> <li>• EASY backfill</li> <li>• Return the List of index of the backfill jobs .</li> </ul>			

<b>Name</b>	backfill_cons			
<b>Input</b>	None	-	-	All the parameters should be stored in the local buffer.
<b>Output</b>	(list)	( list )		List of the backfill jobs. None for no job can be backfill.
<b>Process</b>	<ul style="list-style-type: none"> <li>• Conservative backfill</li> <li>• Return the List of index of the backfill jobs .</li> </ul>			

<b>Name</b>	adapt_reset			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Read the adapt config file and reset the adapt parameter</li> <li>• Add average utilization interval time into <b>Info_collect</b> module.</li> </ul>			

<b>Name</b>	set_adapt_data			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Analysis the information in the <b>Info_collect</b> module and add the new adapt data in the most new item in <b>Info_collect</b> module.</li> </ul>			

<b>Name</b>	get_adapt_info_name			
<b>Input</b>	None	-	-	-
<b>Output</b>	(string)	(string)		The name of the adapt data name in <b>Info_collect</b> module
<b>Process</b>	<ul style="list-style-type: none"> <li>• Return the name of the adapt data name in <b>Info_collect</b> module</li> </ul>			

<b>Name</b>	adapt_read_config			
<b>Input</b>	fileName	(string)	-	Config file name
<b>Output</b>	(int)	(int )		1. success 0. not
<b>Process</b>	<ul style="list-style-type: none"> <li>• Read the adapt config file</li> <li>• Return 1 if success.</li> </ul>			

<b>Name</b>	backfill_adapt			
-------------	----------------	--	--	--

<b>Input</b>	para_in	(list)	-	Current running time parameters
<b>Output</b>	(int)	(int )		1. success 0. not
<b>Process</b>	<ul style="list-style-type: none"> <li>Call the selected adapt method depending on the adapt mode</li> <li>Return 1 if success.</li> </ul>			

<b>Name</b>	adapt_1			
<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	(int )		1. success 0. not
<b>Process</b>	<ul style="list-style-type: none"> <li>Adapt method</li> <li>Return 1 if success.</li> </ul>			

<b>Name</b>	get_list			
<b>Input</b>	inputstring	(string)	-	Input string which need to be analysis into a list
	regex	(string)	r"([^\n]+)"	Regular expression string
<b>Output</b>	(list)	(list )		The result list
<b>Process</b>	<ul style="list-style-type: none"> <li>Analysis the income string and use the income regular expression sample to analysis it.</li> <li>Return the result list of string.</li> </ul>			

<b>Name</b>	get_adapt_list			
<b>Input</b>	None	-	-	-
<b>Output</b>	(list)	(list )		The list of parameters which may be modified when adapt
<b>Process</b>	<ul style="list-style-type: none"> <li>Return the list of parameters which may be modified when adapt</li> </ul>			

### 3.7 Start window

<b>Name</b>	__init__			
<b>Input</b>	mode	(int)	0	Window mode, no difference will be made if only one mode designed.
	ad_mode	(int)	0	Adapt window mode
	node_module	(handle)	None	Node structure module handle
	info_module	(handle)	None	System information module handle
	debug	(handle)	None	Debug module handle
	para_list	(list)	[5,0,0]	Additional parameter list.
	para_list_ad	( list )	None	Additional parameter list for adapt function.
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Initialize the parameters.</li> <li>[win_size] = [para_list[0]]</li> </ul>			

	<ul style="list-style-type: none"> <li>• <code>[check_size_in] = [para_list[1]], [check_size_in] = [win_size]</code> if <code>[para_list[1]]</code> is -1</li> <li>• <code>[start_max_size] = [para_list[2]], [start_max_size] = [win_size]</code> if <code>[para_list[1]]</code> is -1</li> </ul>
--	--

Name	reset			
<b>Input</b>	mode	(int)	None	-
	ad_mode	(int)	None	-
	node_module	(handle)	None	-
	info_module	(handle)	None	-
	debug	(handle)	None	-
	para_list	(list)	None	-
	para_list_ad	( list )	None	-
<b>Output</b>	None	-	/	
<b>Process</b>	<ul style="list-style-type: none"> <li>Reset the parameters.</li> </ul>			

Name	start_window			
<b>Input</b>	wait_job	(list)	-	The list of the related waiting job with the details. Each job information is a dictionary.
	para_in	(dictionary)	None	Running time parameters in the dictionary type.
<b>Output</b>	(list)	( list )	/	
<b>Process</b>	<ul style="list-style-type: none"> <li>This is the entry of the adapt module.</li> <li>Receive the running time information and store them into the local buffers, then invoke <b>main</b> method to deal with the request.</li> <li>Get the reordered job sequence from the <b>main</b> method and return it to the invoker.</li> </ul>			

Name	main			
<b>Input</b>	None	-	-	All the parameters should be stored in the local buffer.
<b>Output</b>	(list)	( list )	/	
<b>Process</b>	<ul style="list-style-type: none"> <li>Provide the adapt function.</li> <li>Return the reordered job sequence .</li> <li>It select different window mode by the input parameter <b>[mode]</b>, and invoke corresponding window method.</li> </ul>			

Name	window_adapt			
<b>Input</b>	para_in	(dictionary)	None	Running time parameters in the dictionary type.
<b>Output</b>	(int)	( int )	/	
<b>Process</b>	<ul style="list-style-type: none"> <li>Change the adapt parameter</li> <li>Return 1 if any parameter changes.</li> </ul>			

<b>Name</b>	window_size			
<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	(int)		Return the window size
<b>Process</b>	<ul style="list-style-type: none"> <li>As the window function only change the order of the first x waiting jobs, it is not necessary for the simulator to pass the whole waiting list into the adapt module.</li> <li>Return the window size. If waiting job list is longer than that, the window module do not care about the rest part.</li> </ul>			

<b>Name</b>	check_size			
<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	(int)		Return the check size
<b>Process</b>	<ul style="list-style-type: none"> <li>Return the check size.</li> </ul>			

<b>Name</b>	start_num			
<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	(int)		The number of the jobs which are started before next window.
<b>Process</b>	<ul style="list-style-type: none"> <li>Return the number of the jobs which are started before next window</li> </ul>			

<b>Name</b>	reset_list			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Reset the buffers and rebuild the sequence list by calling the recursion method <b>build_seq_list()</b>.</li> </ul>			

<b>Name</b>	build_seq_list			
<b>Input</b>	seq_len	(int)	-	Sequence list length
	ele	( list )	-	Element pool in order
	temp_index	( int )	-	The position of the number set in this iteration.
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>This is a recursion method. It keep calling itself until no more element in [ele].</li> <li>In every iteration, the method takes an element out from the element pool.</li> <li>When no more element in the pool, it stop recursion and record all the elements in order, hence a new sequence is produced and be added to the sequence list.</li> <li>This design is to make the sequence list be able to fit different check size in running time: if check size is 3, take first <math>1*2*3=6</math> sequences in the list if check size is 4, take first <math>1*2*3*4=24</math> sequences in the list, and so on.</li> </ul>			

<b>Name</b>	window_check			
<b>Input</b>	None	-	-	-
<b>Output</b>	(list)	( list )		The reordered sequence of the input job list.
<b>Process</b>	<ul style="list-style-type: none"> <li>Do the window check and return the reordered sequence of the input job list.</li> </ul>			

<b>Name</b>	adapt_reset			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Read the adapt config file and reset the adapt parameter</li> <li>Add average utilization interval time into <b>Info_collect</b> module.</li> </ul>			

<b>Name</b>	set_adapt_data			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Analysis the information in the <b>Info_collect</b> module and add the new adapt data in the most new item in <b>Info_collect</b> module.</li> </ul>			

<b>Name</b>	get_adapt_info_name			
<b>Input</b>	None	-	-	-
<b>Output</b>	(string)	(string)		The name of the adapt data name in <b>Info_collect</b> module
<b>Process</b>	<ul style="list-style-type: none"> <li>Return the name of the adapt data name in <b>Info_collect</b> module</li> </ul>			

<b>Name</b>	adapt_read_config			
<b>Input</b>	fileName	(string)	-	Config file name
<b>Output</b>	(int)	(int )		1. success 0. not
<b>Process</b>	<ul style="list-style-type: none"> <li>Read the adapt config file</li> <li>Return 1 if success.</li> </ul>			

<b>Name</b>	window_adapt			
<b>Input</b>	para_in	(list)	-	Current running time parameters
<b>Output</b>	(int)	(int )		1. success 0. not
<b>Process</b>	<ul style="list-style-type: none"> <li>Call the selected adapt method depending on the adapt mode</li> <li>Return 1 if success.</li> </ul>			

<b>Name</b>	adapt_1			
				26 / 52

<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	(int )		1. success 0. not
<b>Process</b>	<ul style="list-style-type: none"> <li>• Adapt method</li> <li>• Return 1 if success.</li> </ul>			

<b>Name</b>	get_list			
<b>Input</b>	inputstring	(string)	-	Input string which need to be analysis into a list
	regex	(string)	r"([^\n]+)"	Regular expression string
<b>Output</b>	(list)	(list )		The result list
<b>Process</b>	<ul style="list-style-type: none"> <li>• Analysis the income string and use the income regular expression sample to analysis it.</li> <li>• Return the result list of string.</li> </ul>			

<b>Name</b>	get_adapt_list			
<b>Input</b>	None	-	-	-
<b>Output</b>	(list)	(list )		The list of parameters which may be modified when adapt
<b>Process</b>	<ul style="list-style-type: none"> <li>• Return the list of parameters which may be modified when adapt</li> </ul>			

### 3.8 Basic Algorithm

<b>Name</b>	__init__			
<b>Input</b>	ad_mode	(int)	0	Adapt mode, no difference will be made if only one mode designed.
	element	(list)	None	Element list of the algorithm.
	info_module	(handle)	None	System information module handle
	debug	(handle)	None	Debug module handle
	ad_para_list	(dictionary)	None	Adapt parameter.
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Initialize the parameters.</li> <li>• Assemble the element list into the algorithm string</li> </ul>			

<b>Name</b>	reset			
<b>Input</b>	ad_mode	(int)	None	-
	element	(list)	None	-
	info_module	(handle)	None	-
	debug	(handle)	None	-
	ad_para_list	(dictionary)	None	-
<b>Output</b>	None	-		-

<b>Process</b>	<ul style="list-style-type: none"> <li>• Reset the parameters.</li> <li>• Assemble the element list into the algorithm string</li> </ul>
----------------	--

<b>Name</b>	get_score			
<b>Input</b>	wait_job	(list)	-	The list of all waiting job with the details. Each job information is a dictionary.
	currentTime	( float )	-	Current virtual time
	para_list	(dictionary)	None	Related system current information.
<b>Output</b>	(list)	( list )		The score list of the wait job. Return None if any error occur.
<b>Process</b>	<ul style="list-style-type: none"> <li>• Receive the job information and system information.</li> <li>• Calculate the job score depending on the input information.</li> <li>• Return the score list.</li> </ul>			

<b>Name</b>	log_analysis			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Look in the local system information log and fill the analysis result of the last record.</li> </ul>			

<b>Name</b>	alg_adapt			
<b>Input</b>	para_in	(dictionary)	None	Running time parameters in the dictionary type.
<b>Output</b>	(string)	(string)		New algorithm string
<b>Process</b>	<ul style="list-style-type: none"> <li>• Receive the current additional parameter.</li> <li>• Check the analysis result and assemble the new algorithm string.</li> </ul>			

<b>Name</b>	adapt_reset			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Read the adapt config file and reset the adapt parameter</li> <li>• Add average utilization interval time into <b>Info_collect</b> module.</li> </ul>			

<b>Name</b>	set_adapt_data			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Analysis the information in the <b>Info_collect</b> module and add the new adapt data in the most new item in <b>Info_collect</b> module.</li> </ul>			

<b>Name</b>	get_adapt_info_name			
<b>Input</b>	None	-	-	-
<b>Output</b>	(string)	(string)		The name of the adapt data name in <b>Info_collect</b> module
<b>Process</b>	<ul style="list-style-type: none"> <li>Return the name of the adapt data name in <b>Info_collect</b> module</li> </ul>			

<b>Name</b>	adapt_read_config			
<b>Input</b>	fileName	(string)	-	Config file name
<b>Output</b>	(int)	(int )		1. success 0. not
<b>Process</b>	<ul style="list-style-type: none"> <li>Read the adapt config file</li> <li>Return 1 if success.</li> </ul>			

<b>Name</b>	alg_adapt			
<b>Input</b>	para_in	(list)	-	Current running time parameters
<b>Output</b>	(int)	(int )		1. success 0. not
<b>Process</b>	<ul style="list-style-type: none"> <li>Call the selected adapt method depending on the adapt mode</li> <li>Return 1 if success.</li> </ul>			

<b>Name</b>	adapt_1			
<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	(int )		1. success 0. not
<b>Process</b>	<ul style="list-style-type: none"> <li>Adapt method</li> <li>Return 1 if success.</li> </ul>			

<b>Name</b>	get_list			
<b>Input</b>	inputstring	(string)	-	Input string which need to be analysis into a list
	regex	(string)	r"([^\n]+)"	Regular expression string
<b>Output</b>	(list)	(list )		The result list
<b>Process</b>	<ul style="list-style-type: none"> <li>Analysis the income string and use the income regular expression sample to analysis it.</li> <li>Return the result list of string.</li> </ul>			

<b>Name</b>	get_adapt_list			
<b>Input</b>	None	-	-	-
<b>Output</b>	(list)	(list )		The list of parameters which may be modified when adapt
<b>Process</b>	<ul style="list-style-type: none"> <li>Return the list of parameters which may be modified when adapt</li> </ul>			

### 3.9 Info collect

<b>Name</b>	<u>__init__</u>			
<b>Input</b>	ave_uti	(list)	None	Average utilization interval list.
	debug	(handle)	None	Debug module handle
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Initialize the parameters.</li> </ul>			

<b>Name</b>	reset			
<b>Input</b>	ave_uti	(list)	None	-
	debug	(handle)	None	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Reset the parameters.</li> </ul>			

<b>Name</b>	info_collect			
<b>Input</b>	time	(float)	-	Virtual time of this information
	event	(int)	-	1:Job, 2:Monitor, 3:Extend, -1:Initial
	uti	(float)	-	The utilization at this time
	waitNum	(int)	-1	Waiting job number at this time
	waitSize	(int)	-1	Total size of all waiting job
	inter	(float)	-1.0	Interval time between this information and next one.
	extend	(list)	None	Other new characters may be added.
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Receive formatted system information and store them as a new item in the list.</li> </ul>			

<b>Name</b>	info_analysis			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Check the new system information and fill the analysis result.</li> </ul>			

<b>Name</b>	get_info			
<b>Input</b>	index	(int)	-	The index of the request information. If it is None, return the whole list.
<b>Output</b>	(dictionary)	( dictionary )		Return the request system information
<b>Process</b>	<ul style="list-style-type: none"> <li>Return the request system information list.</li> <li>Return None if index is exceeded</li> </ul>			

<b>Name</b>	get_len			
<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	(int)		Return the length of the system information list.
<b>Process</b>	<ul style="list-style-type: none"> <li>Return the length of the system information list.</li> </ul>			

<b>Name</b>	calculate_ave_utl			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Calculate the average utilization for the most new system information.</li> </ul>			

<b>Name</b>	reset_utl_interval			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Reset the average utilization interval list</li> </ul>			

<b>Name</b>	reorder_utl_interval			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Reorder the utilization interval list by reset the order list.</li> <li>The order list is for the module to calculate average utilization quicker.</li> </ul>			

### 3.10 Log print

<b>Name</b>	<u>__init__</u>			
<b>Input</b>	filePath	(string)	-	file name
	mode	(int)	0	0: renew file 1: add log
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Initialize the parameters.</li> <li>Call <b>file_open</b> method to open the specified</li> </ul>			

<b>Name</b>	reset			
<b>Input</b>	filePath	(string)	None	-
	mode	(int)	None	-
<b>Output</b>	None	-		-

<b>Process</b>	<ul style="list-style-type: none"> <li>Reset the parameters.</li> </ul>			
----------------	---	--	--	--

<b>Name</b>	file_open			
<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	(int)		1: success 0: Fail
<b>Process</b>	<ul style="list-style-type: none"> <li>Open the specified file.</li> </ul>			

<b>Name</b>	file_close			
<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	(int)		1: success 0: Fail
<b>Process</b>	<ul style="list-style-type: none"> <li>Close the opened file if any file is opened.</li> <li>Return 1 if success, otherwise return 0</li> </ul>			

<b>Name</b>	log_print			
<b>Input</b>	context	(string)	-	Context to print.
	isEnter	(int)	1	1: print Enter after context otherwise: not print enter
<b>Output</b>	(int)	(int)		1: success 0: Fail
<b>Process</b>	<ul style="list-style-type: none"> <li>Print the log to the file specified before.</li> <li>Return 1 if success, otherwise return 0</li> </ul>			

### 3.11 Debug log

<b>Name</b>	__init__			
<b>Input</b>	lvl	(int)	2	0 to 5, 0 is no debug log printed
	show	(int)	2	The lowest level which will be print on the screen.
	path	(string)	None	Debug log path and name.
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Initialize the parameters.</li> </ul>			

<b>Name</b>	reset			
<b>Input</b>	lvl	(int)	None	-
	show	(int)	None	
	path	(string)	None	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Reset the parameters.</li> </ul>			

<b>Name</b>	reset_log			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Clean the specified debug log.</li> </ul>			

<b>Name</b>	set_lvl			
<b>Input</b>	lvl	(int)	0	0 to 4, 0 is no debug log printed, 3 is printing all debug log.
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Reset the debug level.</li> </ul>			

<b>Name</b>	debug			
<b>Input</b>	context	(all type)	-	Debug content, will be changed into string.
	lvl	(int)	3	1 to 4, 0 is no debug log printed, 3 is printing all debug log.
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Call the log print module to add the content to the debug log if [lvl] is smaller than the print log level.</li> </ul>			

<b>Name</b>	line			
<b>Input</b>	lvl	(int)	1	1 to 4, 0 is no debug log printed, 3 is printing all debug log.
	signal	(string)	"_"	Signal to form the line
	num	(int)	15	Duplication number of the signal
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Call the log print module to add the content to the debug log if [lvl] is smaller than the print log level.</li> </ul>			

### 3.12 Output log

<b>Name</b>	__init__			
<b>Input</b>	output	(dictionary)	None	Output file name dictionary
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>• Initialize the parameters.</li> <li>• Initialize all the output file name</li> </ul>			

<b>Name</b>	reset			
<b>Input</b>	output	(dictionary)	None	-
<b>Output</b>	None	-		-

<b>Process</b>	• Reset the parameters.
----------------	-------------------------

<b>Name</b>	reset_output			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-	/	-
<b>Process</b>	• Reset the output file path and name.			

<b>Name</b>	print_sys_info			
<b>Input</b>	sys_info	(dictionary)	-	System information needed to be printed.
<b>Output</b>	None	-	/	-
<b>Process</b>	• Print the current system information to the system information file.			

<b>Name</b>	print_adapt			
<b>Input</b>	adapt_info	( dictionary )	-	Adapt information needed to be printed.
<b>Output</b>	None	-	/	-
<b>Process</b>	• Print the adapt information to the adapt information file.			

<b>Name</b>	print_result			
<b>Input</b>	job_module	( handle )	-	Job trace module
<b>Output</b>	None	-	/	-
<b>Process</b>	• Print all the job result.			

### 3.13 Cqsim sim

<b>Name</b>	<u>__init__</u>			
<b>Input</b>	module	(dictionary)	-	The dictionary of the input module handle.
	monitor	(float)	None	Monitor event time interval.
	debug	(handle)	None	Debug module handle
<b>Output</b>	None	-	/	-
<b>Process</b>	• Initialize the parameters.			

<b>Name</b>	reset			
<b>Input</b>	module	(dictionary)	None	-
	monitor	(float)	None	-

	debug	(handle)	None	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Reset the parameters.</li> </ul>			

<b>Name</b>	cqsim_sim			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>The main process of the simulator.</li> <li>Initialize the event sequence with the job submit event, monitor event and extend event.</li> <li>Scan the event sequence and deal with all the event in the sequence.</li> <li>Output the job result.</li> </ul>			

<b>Name</b>	insert_event_job			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Read the job trace and insert the job submit event in the event sequence in time order.</li> <li>event information:           <ul style="list-style-type: none"> <li>type : 1</li> <li>time: submit time</li> <li>priority: 2</li> <li>para: [1,job index], means this is a submit event.</li> </ul> </li> </ul>			

<b>Name</b>	insert_event_monitor			
<b>Input</b>	start	(float)	-	Start time of the start job
	end	(float)	-	End time of the start job
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Insert the monitor event in the event sequence from <b>[start]</b> to <b>[end]</b>.(Contain <b>[start]</b>, not <b>[end]</b>)</li> <li>event information:           <ul style="list-style-type: none"> <li>type : 2</li> <li>time: monitor time</li> <li>priority: 5</li> <li>para: None</li> </ul> </li> </ul>			

<b>Name</b>	insert_event_extend			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Insert the extend event in the event sequence in time order.</li> <li>event information:</li> </ul>			

	<p>type : 3</p> <p>time: user designed</p> <p>priority: user designed</p> <p>para: user designed</p>
--	--

Name	insert_event			
Input	type	(int)	-	1: job 2: monitor 3: extend
	time	(float)	-	Virtual time of the event
	priority	(int)	-	Priority of the job
	para	(list)	None	Parameter list of the event.
Output	None	-		
Process	<ul style="list-style-type: none"> <li>Insert the event in the sequence, automatically find the place by parameters [time] and [priority].</li> </ul>			

Name	delete_event			
Input	type	(int)	-	1: job 2: monitor 3: extend
	time	(float)	-	Virtual time of the event
	index	(int)	-	The index of the deleting event
Output	(int)	(int)		1: Success 0: Fail
Process	<ul style="list-style-type: none"> <li>Delete the selected event which is indicated by [index] or [time &amp; type]</li> <li>If invoker provides [index] and [time &amp; type], the [index] parameter has higher priority.</li> <li>Return 1 if success, otherwise 0.</li> </ul>			

Name	get_index_monitor			
Input	None	-	-	-
Output	(int)	(int)		Return the current monitor pointer.
Process	<ul style="list-style-type: none"> <li>Return the current monitor pointer.</li> <li>This helps inserting the monitor event</li> </ul>			

Name	scan_event			
Input	None	-	-	-
Output	None	-		-
Process	<ul style="list-style-type: none"> <li>Scan the event sequence recursively.</li> <li>Call the corresponding method to deal with the current event in the sequence, then move the pointer to the event.</li> <li>Stop when no event left in the sequence.</li> </ul>			

Name	event_job			
------	-----------	--	--	--

<b>Input</b>	para_in	(list)	None	Parameter list of the event.
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Deal with the job event (submit/finish).</li> <li>Calculate the scores of the waiting job after the event is done.</li> <li>Call the start scan method group: window - start new job - backfill</li> <li>Store the system information.</li> <li>Insert monitor event from current time to time of the next event.</li> <li>Call the user interface module to show the current system state.</li> </ul>			

<b>Name</b>	event_monitor			
<b>Input</b>	para_in	(list)	None	Parameter list of the event.
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Deal with the monitor event.</li> <li>Call the adapt functions.</li> <li>Call the <b>print_adapt()</b> method if needed.</li> </ul>			

<b>Name</b>	event_extend			
<b>Input</b>	para_in	(list)	None	Parameter list of the event.
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Deal with the extend event.</li> <li>Call the extend process.</li> </ul>			

<b>Name</b>	submit			
<b>Input</b>	job_index	(int)	-	Index of the submitting job.
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Submit the job by calling the corresponding method in <b>job_trace</b> module.</li> </ul>			

<b>Name</b>	finish			
<b>Input</b>	job_index	(int)	-	Index of the finish job.
<b>Output</b>	None	-		-
<b>Process</b>	<ul style="list-style-type: none"> <li>Finish the job by calling the corresponding method in <b>job_trace</b> module.</li> </ul>			

<b>Name</b>	start			
<b>Input</b>	job_index	(int)	-	Index of the finish job.
<b>Output</b>	None	-		-

<b>Process</b>	<ul style="list-style-type: none"> <li>Start the job by calling the corresponding method in <b>job_trace</b> module.</li> <li>Insert job finish event</li> </ul>
----------------	--

<b>Name</b>	score_calculate			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-	/	-
<b>Process</b>	<ul style="list-style-type: none"> <li>Calculate the score for all jobs in waiting list.</li> <li>Reorder the waiting list depending on the score list.</li> </ul>			

<b>Name</b>	start_scan			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-	/	-
<b>Process</b>	<ul style="list-style-type: none"> <li>Scan the jobs in waiting list till no job can be start or backfill.</li> <li>Window function will be used before job start.</li> <li>Backfill function will be used when no job can be started.</li> </ul>			

<b>Name</b>	start_window			
<b>Input</b>	temp_wait_B	(list)	-	Job wait list
<b>Output</b>	(list)	(list)	/	New list after window check
<b>Process</b>	<ul style="list-style-type: none"> <li>Call the window function to modify the order of the waiting job.</li> <li>Return the new reorder list.</li> </ul>			

<b>Name</b>	backfill			
<b>Input</b>	temp_wait	(list)	-	Job wait list
<b>Output</b>	(int)	(int)	/	1: Success 0: no
<b>Process</b>	<ul style="list-style-type: none"> <li>Call the backfill function and get the backfill job list.</li> <li>Start them if any job in the list.</li> <li>Return 1 for some jobs are backfill, 0 for no.</li> </ul>			

<b>Name</b>	sys_collect			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-	/	-
<b>Process</b>	<ul style="list-style-type: none"> <li>Collect the current system information and call the <b>Info_collect</b> module to store them.</li> <li>Print the current system information.</li> </ul>			

<b>Name</b>	interface			
-------------	-----------	--	--	--

<b>Input</b>	sys_info	(dictionary)	None	Current system information need to be shown
<b>Output</b>	None	-		-
<b>Process</b>	• Call the running time user interface module to show the information.			

<b>Name</b>	alg_adapt			
<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	(int)		1: modify 0: not modify
<b>Process</b>	• Call the <b>adapt</b> method in <b>Basic_algorithm</b> module to modify the algorithms in the monitor event process.			

<b>Name</b>	window_adapt			
<b>Input</b>	None	-	-	-
<b>Output</b>	(int)	(int)		1: modify 0: not modify
<b>Process</b>	• Call the <b>adapt</b> method in <b>Start_window</b> module to modify the parameter of window in the monitor event process.			

<b>Name</b>	print_sys_info			
<b>Input</b>	sys_info	(dictionary)	-	System information needed to be printed.
<b>Output</b>	None	-		-
<b>Process</b>	• Print the current system information to the system information file.			

<b>Name</b>	print_adapt			
<b>Input</b>	adapt_info	( dictionary )	-	Adapt information needed to be printed.
<b>Output</b>	None	-		-
<b>Process</b>	• Print the adapt information to the adapt information file.			

<b>Name</b>	print_result			
<b>Input</b>	None	-	-	-
<b>Output</b>	None	-		-
<b>Process</b>	• Print all the job result.			

## 4. Data

### 4.1 Overall

All modules are suppose to know the format of all public data although they do not really know them. Hence, they can get the right data from the incoming dictionary.

So, any change on data format should be record clearly. This section list all public data in every module, the corresponding format are discussed in the next section.

## **4.2 Public Data**

<b>Filter_job</b>	jobList	job data list
	start	start virtual time
	sdate	start date
	density	job submit density modification rate
	anchor	position of the first read job in the original job trace file
	rnum	read job number
	trace	original job trace file name
	save	formatted job trace file name
	jobNum	read job number
	debug	
<b>Filter_node</b>	nodeList	node data list
	struc	original node structure file name
	save	formatted node structure file name
	nodeNum	total node number
	debug	
<b>Job_trace</b>	jobTrace	formatted job data list
	job_submit_list	
	job_wait_list	
	job_run_list	
	job_done_list	
	job_wait_size	
	start	
	read_num	
	density	
	debug	
<b>Node_struc</b>	nodeStruc	formatted node data list
	nodePool	idle node index pool.
	temp_nodePool	
	job_list	running job index list
	predict_node	predict node index
	predict_job	predict job index
	tot	total node number

	idle	idle node number
	avail	max available node number
	debug	
<b>Backfill</b>	para_list_in	
	ad_para_list	
	current_para	
	ad_current_para	
	wait_job	
	para	mode
		ad_mode
		size
		ad_config
	node_module	
	debug	
	adapt_data_name	
	adapt_data_para	
	check_data_name	
	check_data_para	
	ave_utu_interval	
	ave_utu_index	
	adapt_item	
	bound_item	
	adapt_info_name	
<b>Start_window</b>	para_list	
	ad_para_list	
	current_para	
	seq_list	
	temp_list	
	wait_job	
	para	mode
		ad_mode
		win_size
		window size
		check_size
		the first x job will be reorder to find the quickest sequence
		max_start_size
		The max number of job can be start between 2 window function
		ad_config
	node_module	

		info_module	
		temp_check_len	
		debug	
		adapt_data_name	
		adapt_data_para	
		check_data_name	
		check_data_para	
		ave_utl_interval	
		ave_utl_index	
		adapt_item	
		bound_item	
		adapt_info_name	
<b>Basis_Algorithm</b>		para_list	
		ad_para_list	
		scoreList	
	para	mode	
		ad_mode	
		element	algorithm element
		sign	algorithm sign
		ad_config	
		algStr	
		debug	
		adapt_data_name	
		adapt_data_para	
		check_data_name	
		check_data_para	
		ave_utl_interval	
		ave_utl_index	
		adapt_item	
		bound_item	
		adapt_info_name	
<b>Info_collect</b>		sys_info	
		ave_utl_interval	
	order_seq		The index of average utilization interval in order.
		alg_module	
		total_utl	
		debug	
<b>Log_print</b>		modelist	
		filePath	

		mode	
		logFile	
<b>Debug_log</b>	debugFile		
	path		
	lvl		
	show		
<b>Output_log</b>	event_seq		
	sys_info		
	adapt_info		
	job_result		
<b>Cqsim_sim</b>	module		
	event_seq	event sequence list	
	event_pointer	current event index	
	monitor_start	next monitor event index	
	current_event	current event	
	job_num	total job number	
	currentTime		
	monitor	monitor interval time	
	debug		

## 5 Format

### 5.1 User Command Line Format

#### 5.1.1 Cqsim Command Line

5.1.1 Cqsim Command Line						
ID	Name1	Name2	Type	Default	Dest	Comment
1	-j	--job	string	None	job_trace	job trace file name
2	-n	--node	string	None	node_struc	node structure file name
3	-J	--job_save	string	[job trace name]	job_save	formatted job trace data file name
4	-N	--node_save	string	[job trace name] + "_node"	node_save	formatted node structure data file name
5	-f	--frac	float	1	cluster_fraction	job density adjust
6	-s	--start	float	0	start	first job start virtual time
7	-S	--start_date	date	None	start_date	first job start date
8	-r	--anchor	int	0	anchor	first job position in job trace
9	-R	--read	int	-1	read_num	number of jobs read from the job trace

10	-p	--pre	string	"CQSIM_"	pre_name	previous file name
11	-o	--output	string	[job trace name]	output	simulate result file name
12		--debug	string	"debug_"+[job trace name]	debug	debug file name
13		--ext_fmt_j	string	".csv"	ext_fmt_j	formatted job data extension type
14		--ext_fmt_n	string	".csv"	ext_fmt_n	formatted job data extension type
15		--ext_fmt_j_c	string	".con"	ext_fmt_j_c	temp job trace config extension type
16		--ext_fmt_n_c	string	".con"	ext_fmt_n_c	temp job trace config extension type
17		--path_in	string	"InputFiles/"	path_in	input file path
18		--path_out	string	"Reults/"	path_out	output result file path
19		--path_tmp	string	"Temp/"	path_tmp	temp result file path
20		--path_debug	string	"Debug/"	path_debug	debug file path
21		--ext_jr	string	".rst"	ext_jr	job result log extension type
22		--ext_si	string	".ult"	ext_si	system information log extension type
23		--ext_ai	string	".adp"	ext_ai	adapt information log extension type
24		--ext_d	string	".log"	ext_debug	debug log extension type
25	-v	--debug_lvl	int	4	-debug_mode	debug mode
26	-a	--alg	list	None	alg	basic algorithm list
27	-A	--sign	list	None	alg_sign	sign of the basic algorithm element
28	-b	--backfill	int	0	backfill	backfill mode
29	-B	--bf_para	list	None	bf_para	backfill parameter list
30	-w	--win	int-	0	win	window mode
31	-W	--win_para	list	None	win_para	window parameter list
32	-l	--ad_bf	int	0	ad_bf	backfill adapt mode
33	-L	--ad_bf_para	list	None	ad_bf_para	backfill adapt parameter list
34	-d	--ad_win	int	0	ad_win	window adapt mode
35	-D	--ad_win_para	list	None	ad_win_para	window adapt parameter list
36	-g	--ad_alg	int	0	ad_alg	algorithm adapt mode
37	-G	--ad_alg_para	list	None	ad_alg_para	algorithm adapt parameter list
38	-c	--config_n	string	"config_n.set"	config_n	config file - file name and path
39	-C	--config_sys	string	"config_sys.set"	config_sys	system config file
40	-m	--monitor	int	None	monitor	monitor interval time
41	-u	--uti	list	None	ave_utti	average utilization interval list

### 5.1.2 Filter Command Line

5.1.2 Filter Command Line						
ID	Name1	Name2	Type	Default	Dest	Comment
1	-j	--job	string	None	job_trace	job trace file name
2	-n	--node	string	None	node_struc	node structure file name
3	-J	--job_save	string	[job trace name]	job_save	formatted job trace data file name
4	-N	--node_save	string	[job trace name]+"\_node"	node_save	formatted node structure data file name

5	-f	--frac	float	1	cluster_fraction	job density adjust
6	-s	--start	float	0	start	first job start virtual time
7	-S	--start_date	date	None	start_date	first job start date
8	-r	--anchor	int	0	anchor	first job position in job trace
9	-R	--read	int	-1	read_num	number of jobs read from the job trace
10		--ext_tmp_j	string	".csv"	ext_tmp_job	temp formatted job data extension type
11		--ext_tmp_n	string	".csv"	ext_tmp_node	temp formatted job data extension type
12		--ext_tmp_c	string	".con"	ext_tmp_config	temp job trace config extension type
13		--path_in	string	"Input Files/"	path_in	input file path
14		--path_tmp	string	"Temp/"	path_tmp	temp result file path
15	-c	--config_n	string	"config_n.set"	config_n	config file - file name and path

## 5.2 Basis Algorithm Format

The basic algorithm use some simple letters to represent the different informations of a job. The algorithm method stores the information in these buffers and then calculate the scores with them.

s	Job submit time
t	Job estimated running time
n	Job required nodes #
w	Job waiting time
m	Current idle nodes #
l	Longest job estimated time (in waiting list)
z	Longest job waiting time (in waiting list)

The structure of the algorithm string is stored as [elements of the algorithm string, the signal of the element] pairs in a list.

element	A string contain the element.
signal	1: The element will be changed in future 0:The element will not be changed in this simulator

For example the algorithm list is:

"0.75"	"* w/z+"	"0.25"	"*l/t"
1	0	1	0

So, the algorithm string is "0.75\* w/z+0.25\*l/t " and the elements will be changed in future are "0.75" and "0.25".

## 5.3 Job Trace Format

The type of the job trace is list of dictionary.

Dictionary	Type	Comment	Initial
------------	------	---------	---------

Name	Type	Comment	
id	int	The id of the job	-1
submit	float	Submit time of the job	-1.0
wait	float	Actual waiting time	-1.0
run	float	Actual running time	0.0
usedProc	int	Actual processes the job takes	0
usedAveCPU	float		0.0
usedMen	float	Actual used memory	0
reqProc	int	The processes required by user	0
reqTime	float	The running time required by user	0.0
reqMem	float	Kilobytes per processor	0.0
status	int	Status of the job	0
userID	int	User ID	-1
groupID	int	Group ID	-1
num_exe	int	Executable number	0
num_queue	int	Queue number	0
num_part	int	Partition number	0
num_pre	int	Preceding job number	0
thinkTime	int	Think time from preceding job	0
start	float	Job start time	-1.0
end	float	Job end time	-1.0
score	int	Job scores, shows the priority of the job	0
state	int	0: Not submit, 1:In waiting list, 2:Running, 3:Done	0
happy	int	0: Not happy, 1:Happy, -1:Not care	-1
estStart	float	Estimated start time, the time predicted to run when the job is submitted considering no backfill or any other modification in job order.	-1.0
extend	list	Other new characters may be added.	None

## 5.4 Node Structure Format

The type of the node structure is list of dictionary.

Dictionary Name	Type	Comment	Initial
id	int	The id of the node.	-1
location	list	The location of the node, kind of [x,y,z] or [x,y]. Can also be None if you do not care about the location of the node.	None
group	int	Group ID of the node.	1
state	int	-1: Idle, Other: The index of the job which takes the node	-1
proc	int	Processes number in the node.	1

start	float	Start time of the occupy of the node.	-1
end	float	Estimated end time of the occupy of the node.	-1
extend	list	Other new characters may be added.	None

The type of the predict node structure is list of dictionary.

Dictionary Name	Type	Comment	Initial
time	float	Time of the event take place	-
idle	int	Idle process number	-
avail	int	Available process number	-

The type of the predict job structure is list of dictionary.

Dictionary Name	Type	Comment	Initial
job	int	Job index	-
start	float	Job estimate start time	-
end	float	Job estimate end time	-

## 5.5 Event Sequence Format

The type of the node strucutre is list of dictionary.

Dictionary Name	Type	Comment	Initial
type	int	1:Job, 2:Monitor, 3:Extend, -1:Initial	-1
time	float	Virtual time when the event takes place	-1.0
priority	int	Priority of the event, higher priority will take place earlier if there is another event at the same time.	5
para	list	Parameter list which will be transferred into the corresponding method.  Job event: submit: [1, job index] (Q) start: [3, job index] (S) finish: [2, job index] (E)	None

## 5.6 System Information Format

The type of the system information is list of dictionary. It will make a record when an event takes place (event here includes job start).

Dictionary Name	Type	Comment	Initial
date	date	Date of the event	None

time	float	Virtual time of this information	-1.0
inter	float	Interval time between this information and next one.	-1.0
uti	float	The utilization at this time	-1.0
waitNum	int	Waiting job number at this time	-1
waitSize	int	Total size of all waiting job	-1
event	string	‘Q’: submit ‘S’: start ‘E’: end ‘C’: monitor	None
tot_ave_uti	float	Overall average utilization	0.0
ave_uti	list	Average utilization list in order.	[]
extend	list	Other new characters may be added.	None

## 5.7 Config File Format

Every line contains a data: [data name]=[data value]

No rest space should appeared in the line. If there are some spaces, the regular expression function will not take them as “useless” signal, sp some error may occur because the system can not transform the space into a number or can not find the file because of the addition space.

Or you may want to add some codes to ignore the additional space. But these codes are not there now.

### 5.7.1 File Name And Path Config File

Name	Type	Comment
pre_name	string	previous file name
ext_fmt_j	string	formatted job data extension type
ext_fmt_n	string	formatted job data extension type
ext_fmt_j_c	string	formatted job trace config extension type
ext_fmt_n_c	string	formatted node structure config extension type
path_in	string	input file path
path_out	string	output result file path
path_tmp	string	temp result file path
path_debug	string	debug file path
ext_jr	string	job result log extension type
ext_si	string	system information log extension type
ext_ai	string	adapt information log extension type
ext_debug	string	debug log extension type

### 5.7.2 System Parameter Config File

Name	Type	Comment
cluster_fraction	float	job density adjust
start	float	first job start virtual time
start_date	date	first job start date
anchor	int	first job position in job trace

read_num	int	number of jobs read from the job trace
debug_lvl	int	debug level
alg	list	basic algorithm list
alg_sign	list	sign of the basic algorithm element
backfill	int	backfill mode
bf_para	list	backfill parameter list
win	int	start window mode
win_para	list	start window module parameter: [window size],[check size],[max start size],[max window size]
ad_win	int	start window adapt mode
ad_bf	int	backfilladapt mode
ad_alg	int	algorithm adapt mode
ad_win_para	list	adapt start window parameter list It contains the config file name
ad_bf_para	list	adapt backfill parameter list It contains the config file name
ad_alg_para	list	adapt basic algorithm parameter list It contains the config file name
config_n	string	config file - file name and path
monitor	float	interval time of monitor event
ave_utti	list	The interval list of the average utilization. This parameter will be transmitted into the info_collect module.
job_trace	string	job trace file name
node_struc	string	node structure file name

### 5.7.3 Adapt Config File(Basic Algorithm/Backfill/Start Window)

Name	Type	Comment
adapt_data_name	list	Adapt data name in order: names should not combine if they are same. <b>example:</b> [name 1],[name 2],...,[name X]
adapt_data_para	list	Adapt data parameter in order: In most time, the parameter is the index of the corresponding data, -1 mean this data is not a list. This is really depend on the design of the config file reading method. <b>example:</b> [para 1],[ para 2],...,[ para X]
check_data_name	list	The name of data need to be check when adapt. <b>example:</b> [name 1],[name 2],...,[name Y]
check_data_para	list	The parameter of data need to be check when adapt. <b>example:</b> [para 1],[ para 2],...,[ para Y]
ave_utti	list	Average utilization interval list.

		This list contain all the average utilization need to be check								
adapt_item	list	<p>This is the main part of the adapt function.</p> <p>One <b>adapt_item</b> can be add if you want a new adapt choice.</p> <p>For example, you can add an <b>adapt_item</b> to indicate that the <i>i</i>th data in <b>adapt_data_name</b> will -1 if the data in <b>check_data_name</b> is in case A. And add another <b>adapt_item</b> to indicate that the <i>i</i>th data in <b>adapt_data_name</b> will +1 if the data in <b>check_data_name</b> is in case B. Hence the <i>i</i>th data can be modified in running time depending on different cases of the check data.</p> <p>One thing needed to be mentioned is you can add conflicted case, but the function will only choose the first one who satisfy the request.</p> <p>All <b>adapt_item</b> should be written in the right format:</p> <p>All data should be written as in a list separating by "", without no addition space.</p> <table border="1"> <tr> <td>0</td><td>Index of the data in <b>adapt_data_name</b></td></tr> <tr> <td>1</td><td>0: change the adapt data to the next value 1: add the next value to the adapt data</td></tr> <tr> <td>2</td><td>The new value will be set to/ad to the corresponding adapt data.</td></tr> <tr> <td>3~(3+2*Y+1)</td><td>Y is the number of check data – 1. These data indicate the case request. For jth check data, it can be considered as “in the case” if <b>adapt_item[3+j*2]≤check_data[j]&lt;adapt_item[3+j*2+1]</b> If all the check data is in the case, then this adapt item is the right one.</td></tr> </table>	0	Index of the data in <b>adapt_data_name</b>	1	0: change the adapt data to the next value 1: add the next value to the adapt data	2	The new value will be set to/ad to the corresponding adapt data.	3~(3+2*Y+1)	Y is the number of check data – 1. These data indicate the case request. For jth check data, it can be considered as “in the case” if <b>adapt_item[3+j*2]≤check_data[j]&lt;adapt_item[3+j*2+1]</b> If all the check data is in the case, then this adapt item is the right one.
0	Index of the data in <b>adapt_data_name</b>									
1	0: change the adapt data to the next value 1: add the next value to the adapt data									
2	The new value will be set to/ad to the corresponding adapt data.									
3~(3+2*Y+1)	Y is the number of check data – 1. These data indicate the case request. For jth check data, it can be considered as “in the case” if <b>adapt_item[3+j*2]≤check_data[j]&lt;adapt_item[3+j*2+1]</b> If all the check data is in the case, then this adapt item is the right one.									
bound_item	list	<p>This is similar to the <b>adapt_item</b>. It defines the bound of all the adapt data when you add the new value in adapt item to them.</p> <table border="1"> <tr> <td>0</td><td>Index of the data in <b>adapt_data_name</b></td></tr> <tr> <td>1</td><td>Smallest value</td></tr> <tr> <td>2</td><td>Biggest value</td></tr> </table>	0	Index of the data in <b>adapt_data_name</b>	1	Smallest value	2	Biggest value		
0	Index of the data in <b>adapt_data_name</b>									
1	Smallest value									
2	Biggest value									

## 5.8 Formatted File

2 kinds of formatted file(job/node data temp file) have the same structure:

- Each item takes a single line. For each line, the data are stored in the order

which is described in previous section(white part). Every single data in the extend part should be store as a single data.

data 1	data 2	data 3	...another data
--------	--------	--------	-----------------

- “;” is used as the separated signal in a line. “\n” are used to separate lines.

Formatted config file:

- Each value takes a single line: [data name]=[data value]
- No additional space

## 5.9 Parameters Format

### 5.10.1 wait\_job (Method: backfill Class: Backfill)

wait_job (Method: backfill Class: Backfill)		
Name	Type	Comment
index	int	Job index
proc	int	Request processes number
node	int	Request nodes number
run	float	Request running time
score	float	Job score

## 5.10 Output Format

The output data separated with “;”.

### 5.10.1 Job result

Name	Comment
ID	Job ID (not index)
Request process	Request processes number
Request node	Request nodes number
Request time	Request time
Run	Run time
Wait	Waiting time
Submit	Job submit time
Start	Job start time
End	Job finish time
Node list	The nodes which the job take

### 5.10.2 Event Log

Name	Comment
Wall clock time	Date + Time

Event type	Q: submit S: job start E: job end C: monitor
Virtual event time	virtual event time
Other parameter	[data name 1]=[data value 1] [data name 2]=[data value 2] ... Different data separate by space

Other parameter in Event Log	
Name	Value
uti	System utilization
waitNum	Wait job number at that time
waitSize	Wait job total size

### 5.10.3 Adapt Log

Name	Comment
Virtual time	virtual time
Start window adapt data	Start window adapt data, separated with “;”
Basic algorithm adapt data	Basic algorithm adapt data, separated with “;”
Backfill adapt data	Backfill adapt data, separated with “;”

## 5. Extension

### 3.1 Overall

The program is designed to be an extendable one. All module except **Info\_collect**\* can be modified to fit new request with keeping the port same.

\*In order to keep **Start\_window**, **Basic\_algorithm** and **Backfill** module independent and efficient, they know the inside structure of the **Info\_collect** module. So you need to modify the three module when you modify the **Info\_collect** module.

It can be extended in 3 ways:

1. Input different parameters or new data in config files
2. Build new subclass of modules to fit special request. You should invoke the subclass and change the object define in **Cqsim\_main.py**
3. Modify the original code. You should make this kind of modification only when it fit all application. For example, you may want to modify the original code when you need to trace more running time information, the additional information is useful in most case and you can easily choose not to track them if you don't need.

You should make sure that all related parts know that change and call the new function in the right way. This is easy to implement because the modules are all highly independent.